Chapter TWENTY Assertions

Exam Objectives

Test invariants by using assertions.

Using assertions

An assertion is a statement used to check if something is true and helps you to detect errors in a program.

An assert statement has the following syntax:

```
assert booleanExpression;
```

If `booleanExpression` evaluates to **FALSE**, an exception of type `java.lang.AssertionError` (a subclass of `Error`) is thrown.

In other words, the above statement is equivalent to:

```
if(booleanExpressionIsFalse) {
     throw new AssertionError();
}
```

This is because, as said before, an assertion has to be true. If it's not, an error has to be thrown.

An assert statement can also take a `String` as a message:

```
assert booleanExpression: "Message about the error";
```

Which is equivalent to:

```
if(booleanExpressionIsFalse) {
     throw new AssertionError("Message about the error");
}
```

The thing is, assertions are **NOT** enabled by default.

You can have assertion all over you code, but if they are disabled, Java will skip them all.

In summary:

- If assertions are enabled and the `boolean` expression is true, nothing happens.
- If assertions are enabled and the `boolea` n expression is false, an `AssertionError` is thrown.
- If assertions are disabled, no matter the outcome of the `boolean` expression, assertions are ignored.

Assertions are enabled in the command-line with either:

```
java -ea MainClass
```

Or

```
java -enableassertions MainClass
```

This would enable assertions in all the classes of our program except for the classes of Java (or system classes).

For example, if we have this class:

```
public TestAssertion {
     public static void main(String[] args) {
         // Parentheses are optional
        assert (args.length > 0) :
             "At least one argument is required";        System.out.println(args[0]);
}
```

And we run it this way:

```
java -ea TestAssertion
```

The output will be:

```
Exception in thread "main" java.lang.AssertionError: At least one argument is required
```

If we run it this way:

```
java -ea TestAssertion Hi
```

The output will be:

```
Hi
```

If we run it this way:

```
java TestAssertion
```

The output will be:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0 at com.example.T
```

As you can see, assertions can help us catch some errors derived from assumptions we make in our program. Just be aware that they are not enabled by default. They are not designed to be run in production, for that, it's better to use regular exceptions.

Assertions are better used to validate parameters, properties, preconditions, postconditions, application flow of control (for points of the code that should never be reached), and in general, error checking that otherwise, you would have to comment or disabled when the code is ready for production.

Assertions can also be enabled for one class:

```
java -ea:ClassName MainClass
```

Or for all classes in a package:

```
java -ea:com.example MainClass
```

For enabling assertions in the unnamed package (when you don't use a package statement):

```
java -ea:... MainClass
```

Or in the rare case you'd want to enable assertions for the system classes:

```
java -esa MainClass
```

Or

```
java -enablesystemassertions MainClass
```

There's also a command to disable assertions:

```
java -da MainClass
```

Or

```
java -disableassertions MainClass
```

With the same options that `-ea`. To disable for one class:

```
java -da:ClassName MainClass
```

For disabling assertions in a package:

```
java -da:com.example MainClass
```

For enabling assertion in the unnamed package:

```
java -ea:... MainClass
```

For the system classes:

```
java -dsa MainClass
```

Or

```
java -disablesystemassertions MainClass
```

This option exists because there will be times when you want to use commands like:

```
java -ea -da:ClassOne MainClass
```

To enable assertions in all the program classes except for `Class1` .

Key Points

- An assertion is a statement used to check if something is true and helps you to detect errors in a program.
- An assert statement has the following syntax:
  ```
  assert booleanExpression;
  assert booleanExpression: "Message about the error";
  ```
- If the `boolean` expression of the assert statement evaluates to `false` , the program will throw a `java.lang.AssertionError` and terminate.
- By default, assertions are disabled. You can use the command-line arguments `-ea` (for enabling assertions) and `-da` (for disabling assertions) with other options for classes, packages, and system classes when running the program.

Self Test

1. Given:

```
public class Question_20_1 {
    public static void main(String[] args) {
        int x = 0;
        assert ++x > 0 : x;
    }
}
```

What is the result when this program is executed with assertions enabled?
A. `0`
B. `1`
C. Nothing is printed
D. An exception is thrown at runtime

2. Which of the following statements is true?
A. You can disable assertion at the command-line.
B. Assertions are enabled by default in system classes.
C. Even if assertions are disabled, if the `boolean` expression evaluates to false, a runtime error is thrown.
D. Depending on the syntax, assertions can throw either a checked or a runtime exception.

3. Given:

```
java -esa -ea:com.example MainClass
```

Which of the following statements is true?
A. This command enables assertions in all classes of our program.
B. This command enables assertions in system classes in all classes of our program.
C. This command enables assertions in system classes in just `MainClass`.
D. This command enables assertions in system classes of the packages `com.example`.

4. Given:

```
public class Question_20_4 {
    public static void main(String[] args) {
        assert isValid();
    }
    public static boolean isValid() {
        return false;
    }
}
```

What is the result when this program is executed with assertions enabled?
A. `false`
B. Nothing is printed
C. A compile time error
D. An `AssertionError` is thrown at runtime

5. Which of the following statements is equivalent to:

```
assert false : "Assertion is false";
```

A. `throw new Assertion("Asssertion is false");`
B. `throw new AssertionError("Asssertion is false");`
C. `if(false) throw new AssertionError("Asssertion is false");`
D. `if(false) throw new RuntimeException("Asssertion is false");`

Open answers page

---

19. Exceptions 21. Core Date/Time Classes